Devoir de Programmation PC2R 2017 - Jeu de Lettre

Version du 27/02

Description du devoir

But du devoir

Le but du devoir est de réaliser une application clients-serveur permettant à des utilisateurs de jouer à un jeu de lettre multijoueurs (type *Scrabble Duplicate*[1]): on fournit à tous les participants connectés l'état du plateau de jeu (des cases éventuellement remplies de lettres) et le même tirage de 7 lettres; les participants envoient des propositions de placement des lettres sur le plateau; à la fin du tour, tous les participants marquent des points en fonction de leur placement et le jeu continue pour tous les participants avec le plateau obtenu apres le meilleur placement proposé.

Exigences techniques

L'application doit être réalisée suivant une architecture client / serveur.
La partie client et la partie serveur doivent être chacune écrite dans un langage différent, les deux langages appartenants à l'ensemble {`C`, `Java`, `OCaml`}.
L'utilisation d'autres langages est possible mais soumise à autorisation de la part de l'équipe pédagogique.
Le client et le serveur doivent pouvoir fonctionner indépendamment l'un de l'autre (et pouvoir fonctionner avec le client et le serveur de l'équipe pédagogique). Ils doivent respecter le protocole décrit plus loin.

Règles du jeu

Déroulement d'une session de jeu

Au moins 1 joueur participe à une session de jeu. Il n'y a, a priori, pas de nombre maximum de joueurs.
Une session de jeu comprends plusieurs tours de jeu. Le score d'un joueur persiste au cours d'une session. Le plateau de jeu est vide au début d'une session et se remplit (de lettres) au cours des tours de la session. La session s'arrête quand toutes les lettres ont été tirées, ou quand le nombre de joueurs descend à 0.
Un joueur peut rejoindre ou quitter une session à tout moment. Quand un joueur rejoint le jeu au milieu d'un tour, il peut directement participer au tour courant.
Un tour de jeu se déroule ainsi:

- **Début du tour:** Le serveur diffuse à tous les participants l'état courant du plateau de jeu 15x15, c'est à dire le contenu des 225 cases du plateau (qui peuvent être soit vides, soit contenir une lettre) et le tirage courant de 7 lettres.

- **Phase de recherche:** Un compte à rebours long (5 min) s'active apres cette diffusion, pour permettre aux joueurs de réfléchir à un placement. A tout moment, un joueur peut envoyer un placement au serveur, sous la forme de la description de ce plateau apres le placement (le contenu des 225 cases). Le serveur vérifie que le placement est valide. Si c'est le cas, la phase de recherche s'arrête. Si personne n'a proposé de placement valide à la fin du compte à rebours, le tour s'arrête et on commence un nouveau tour avec le même plateau et un nouveau tirage de lettres.

- **Phase de soumission:** Un compte à rebours court (2 min) s'active à la première soumission valide. Le serveur signale aux joueurs que l'on passe dans la phase de soumission. Tous les joueurs peuvent envoyer des propositions, le serveur leur signale si elle est valide ou non et, dans le premier cas, calcule le score de la proposition

[1] cf. Section "Ressources"

(basé sur les lettres utilisées et leur placement). Un même joueur peut envoyer plusieurs propositions; le serveur garde en mémoire, pour chaque joueur, la proposition valide réalisant le score le plus haut.

- **Phase de résultat:** Les résultats du tour (score de tous les joueurs) sont diffusés aux participants ainsi que le mot ayant réalisé le score maximum. Apres un délai tres court (10s), un nouveau tour commence avec comme plateau de jeu, le plateau résultant du placement du meilleur mot, et un nouveau tirage de lettres.

Regles du jeu de lettre

Le **tirage** des 7 lettres à chaque tour doit comporter une part d'aléatoire et doit converger (on ne peut faire qu'un nombre fini de tirages, puis c'est la fin de la session). La manière classique de procéder est de disposer d'un même réservoir de lettre au début de chaque session. A chaque tour de la session, on tire (sans remise) dans le réservoir autant de lettres que celles qui ont été utilisées au tour précédent (et on garde les lettres non utilisées). Si aucun mot n'a été trouvé au tour précédent, on jette les 7 lettres et on en pioche 7 nouvelles. D'autres manières de procéder sont possibles.

Le **placement** des lettres par les joueurs doit obéir à certaines règles. De manière classique, on requiert que les lettres soit placées de telles facons qu'elles forment un mot utilisant une lettre déjà présente sur le plateau (sauf au premier tour), en coupant perpendiculairement un mot déjà présent. De manière classique, on pourra aussi compléter un mot déjà existant en un nouveau mot valide en ajoutant des lettres comme préfixe ou suffixe de ce mot. Enfin, dans le cas ou le mot est placé à côté d'un mot qui lui est parallèle, tous les mots perpendiculaires créés doivent être valides. D'autre règles de placement sont possibles.

Un **mot** est une suite de lettres valides, c'est à dire présentes dans un dictionnaire sur le serveur. Ce devoir nécessite donc de mettre à disposition un dictionnaire de mots côté serveur. Il existe des dictionnaires libres qui peuvent être récupérés sur Internet (par exemple le GLAFF `http://redac.univ-tlse2.fr/lexiques/glaff.html`, ou encore `http://www.pallier.org/ressources/dicofr/dicofr.html`) et la création d'un dictionnaire personnel est possible. Une extension facultative proposée (cf. plus bas) est l'écriture d'un serveur qui vérifie les mots dynamiquement sur le Web (plutôt qu'avec un dictionnaire interne).

Le **score** d'un mot doit dépendre des lettres du mot. La manière classique est d'attribuer à chaque lettre du réservoir une valeur en point, et de considérer le score d'un mot comme la somme de ces points. Eventuellement, on peut faire entrer en jeu des critères de position sur le plateau (cases "mot compte triple", "lettre compte double"). Le score d'un placement correspond à la somme des scores des nouveaux mots créés par le placement.

Protocole

Le serveur peut être initialisé avec de nouvelles valeurs pour les compte-à-rebours et éventuellement un fichier cible pour le dictionnaire. Le serveur écoute sur le port 2017 en TCP. Clients et Serveur échangent selon un protocole texte. Une commande est composée de chaînes ASCII terminées par des `/`. La commande elle-même est terminée par un `\n`. En cas de doute sur la non-ambguité ou la complètude du protocole, écrire à l'équipe pédagogique.

Tous les protocoles sont **extensibles** si nécessaire (notamment pour le traitement des extension). Tous les clients et les serveurs doivent gérer raisonnablement les extensions de protocoles non-reconnues (par exemple, en ignorant les commandes non reconnues). Par exemple, les clients-serveurs doivent être résilients face à une méthode de calcul de score différente entre le client et le serveur (c'est le score du serveur qui prime, évidemment).

Connexion

```
CONNEXION/user/
(C -> S) Nouvelle connexion d'un client nomme 'user'
BIENVENUE/placement/tirage/scores/
(S -> C) Validation de la connexion. Envoi du plateau courant, tirage courant et scores.
REFUS/
(S -> C) Refus de la connexion (par exemple parce qu'un client avec le meme nom est deja connecte).
CONNECTE/user/
(S -> C) Signalement de la connexion de 'user' aux autres clients.
```

Déconnexion

```
SORT/user/
(C -> S) Déconnexion de 'user'.
DECONNEXION/user/
(S -> C) Signalement de la deconnexion de 'user' aux autres clients.
```

Début d'une session

```
SESSION/
(S -> C) Début d'une session.
VAINQUEUR/bilan/
(S -> C) Fin de la session courante, scores finaux de la session.
```

Phase de recherche

```
TOUR/plateau/tirage/
(S -> C) Début d'un nouveau tour, plateau courant et tirage courant.
TROUVE/placement/
(C -> S) Annonce d'une solution de placement par un joueur.
RVALIDE/
(S -> C) Validation de la solution par le serveur, fin de la phase de recherche.
RINVALIDE/raison/
(S -> C) Invalidation de la solution par le serveur, raison explicite.
RATROUVE/user/
(S -> C) Signalement d'un mot trouve par 'user'. Fin de la phase de rech. et début de la  phase de soum.
RFIN/
(S -> C) Expiration du delai imparti a la reflexion. Fin de la phase de recherche et nouveau tour.
```

Phase de soumission

```
TROUVE/placement/
(C -> S) Annonce d'une solution de placement par un joueur.
SVALIDE/
(S -> C) Validation de la solution par le serveur.
SINVALIDE/raison/
(S -> C) Invalidation de la solution par le serveur, 'raison' explicite.
SFIN/
(S -> C) Expiration du delai imparti a la soumission, fin de la phase de soumission et phase de resultat.
```

Phase de résultat

```
BILAN/mot/vainqueur/scores/
(S -> C) Bilan du tour, nom et mot du gagnant, scores de tous les joueurs.
```

Il faut noter que `TROUVE` existe dans deux phases pour des raisons d'**asynchronie**: en effet, il est nécessaire que le serveur prenne en compte une proposition qui a été faite pendant la phase de recherche et arrive pendant la phase de soumission.

Chaînes

- `user` est une chaîne de lettres identifiant un joueur.

- `placement` décrit l'état d'un plateau de jeu. Elle est utilisée par le serveur pour diffuser l'état courant et par les joueurs pour proposer un placement. Elle décrit l'état des 255 (15x15) cases du plateau par une chaine de 255 caracteres décrivant le contenu de chaque case, lignes par lignes, de la gauche vers la droite, en partant de la ligne la plus haute. Si une lettre est présente sur une case, le caractère utilisé est la lettre majuscule, si la case est vide, le caractère est `0`.
- `tirage` décrit un tirage de septs lettres et est composé de 7 caractères comportant les 7 lettres tirées en majuscule.
- `raison` décrit la raison pour laquelle un placement est refusé. Les chaines à envoyer sont libres (on peut être aussi expicite que possible) mais doivent commencer par 'POS' (position des lettres invalides), 'DIC' (position des lettres valides mais mot qui n'existe pas dans le dictionnaire) ou 'INF' (mot valide mais score inférieur à un précédent mot proposé par le même joueur).
- `scores` décrit l'état des scores pour la session courante sous la forme `n*user1*score1*user2*score2*...` où `n` est le nombre de tours écoulés dans la session, `useri` parcourt le noms des joueurs et `scorei` leurs scores.

Extensions

Il est probable que la réalisation d'extension(s) implique un enrichissement du protocole. Il est important que, au maximum, clients et serveurs bénéficiant d'extensions soient compatibles avec ceux n'en bénéficiant pas. En dernier recours, il est possible de rendre deux versions des clients et serveurs ("compatible" et "enrichie").

Extensions nécessaires

Pour être bien évalué, le projet doit implanter les extensions suivantes:

- **Chat:** Le client et les serveurs gerent un minisalle de clavardage qui permet aux clients de communiquer entre eux à tout moment (quelque soit la phase de jeu)

  ```
  ENVOI/message/
  (C -> S) Envoi (public) d'une chaine de caractere "message" à tous les joueurs.
  PENVOI/user/message/
  (C -> S) Envoi (privé) d'une chaine de caractere "message" au joueur "user" uniquement.
  RECEPTION/message/
  (S -> C) Réception d'un message public.
  PRECEPTION/message/user/
  (S -> C) Réception d'un message prive de l'utilisateur "user".
  ```

- **Meilleur mot:** Le client dispose d'un booléen (affiché à l'écran sous forme d'un voyant, ou de la couleur de l'interface), qui lui permet de savoir si le mot qu'il a proposé est le meilleur (mais il ne donne pas d'autres informations). Bien sur, ce booléen change **en temps réel**, en fonction des propositions des autres joueurs. La pertinence et l'efficacité de l'utilisation des messages liés à ce booléen sera évaluée.

  ```
  MEILLEUR/statut
  (S -> C) Indique au client que son mot est le meilleur mot proposé ("1") ou non ("0").
  ```

Extensions facultative

Pour obtenir la note maximale, le projet doit implanter des extensions supplémentaire parmi les suivantes (ou d'autres extensions non prévues dans ce sujet):

- **Dictionnaire distant**: acces a un dictionnaire sur le web. Le serveur ne stocke pas les mots valides mais fait appel à un serveur web distant (existant ou programmé par les étudiants) qui i) contient un dictionnaire ou ii) teste l'existence d'un mot.

- **Persistance**: système d'authentification pour les clients, mise en mémoire des statistiques des parties précédentes, possibilité de rejoindre une partie quittée précédemment.
- **Journal**: publication régulière des résultats des dernières parties sur une page web.
- **Client autonome**: programmation d'un client tricheur qui calcule automatiquement une solution (voire la meilleure). C'est encore mieux si le tricheur peut se faire passer pour un humain (comportement crédible, messages de chat).
- **Variantes**: implantations de variantes (existantes ou inédites) du jeu.

Points d'intérêt

L'évaluation d'un projet s'intéressera (entre autres) aux points suivants:

- **Qualité de la gestion de la concurrence**: le serveur doit gérer plusieurs threads clients qui accedent de manière **sûre** et **efficace** à des ressources partagées.
- **Respect du protocole** et compatibilité avec les clients/serveurs des autres groupes.
- **Qualité du Rapport**: description et justifcation des choix d'implémentation et manuel du jeu.
- **Jouabilité**: ergonomie de l'interface client, et qualité des entrées/sorties (paramètres, journal) du serveur.

Ressources

- Wikipedia sur le *Scrabble Duplicate*: `https://fr.wikipedia.org/wiki/Scrabble_duplicate`
- Exemple de version Web du jeu: `https://duplika.ca/`